CVE-2018-1000224

Godot serialization security issues

Luiz Felipe Moumdjian Girotto
Victor Chiaradia Gramuglia Araujo

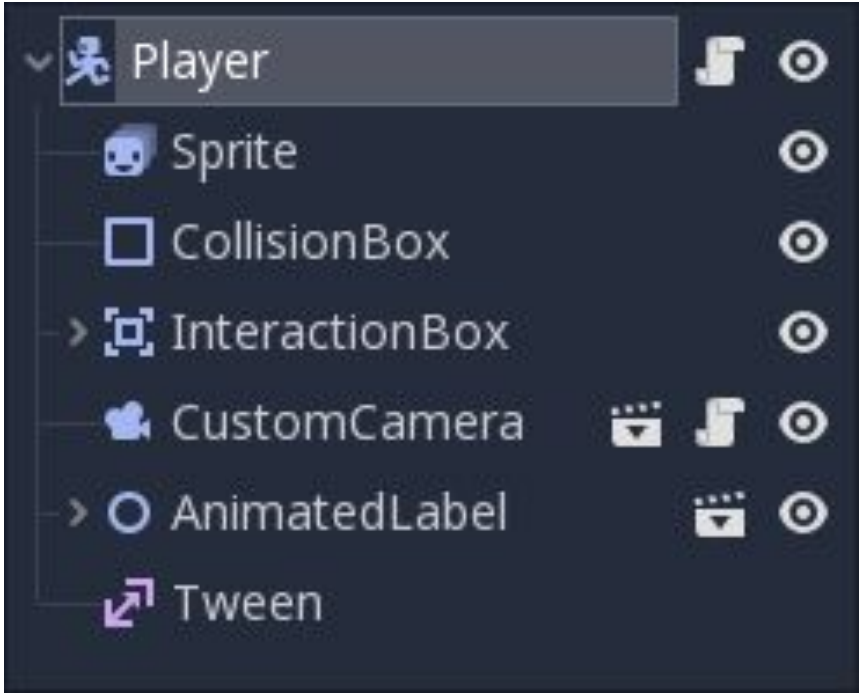
Introdução

Godot

- Uma *game engine*.
- Criada em 2007, primeiro *release* e *open source* em 2014.
- Versão 2.0 em 2016, e 3.0 em 2018, ambas mantidas até hoje.

Godot

- Utilizada para criar jogos 2D e 3D.
- Jogos construídos a partir de **nós** e **cenas.**



Godot e *GDScript*

- Suporta oficialmente C, C++, *VisualScript*, e *GDScript*.

- *GDScript*: alto nível, tipagem dinâmica, alta integração com os elementos e estruturas da *engine*.

GDScript

- Possui *Variant* como tipo de dados atômico/nativo.
- Todos outros tipos de dados *built-in* na linguagem dependem da existência do tipo *Variant*.

O Problema

Godot e *Networking*

- Godot disponibiliza APIs para a criação de jogos multijogador *online*.
- Disponibiliza APIs de baixo nível (TCP e UDP), mas também disponibiliza de alto nível.

Networking

- A transmissão de dados por via destes protocolos de baixo nível requer a serialização (ou *marshalling*) dos dados a serem enviados.

- Problemas no código de serialização até as versões 2.1.5 e 3.0.6 da Godot.

Código de Serialização

- Problemas de *padding* causando o vazamento de memória interna pela rede.

- Problemas na checagem de tamanho de certos tipos de dados levam a *engine* a tentar alocar grandes quantidades de memória, e ser fechada pelo SO.

Código de Serialização

```
case Variant::POOL_BYTE_ARRAY: {

        ERR_FAIL_COND_V(len < 4, ERR_INVALID_DATA);
-       uint32_t count = decode_uint32(buf);
        buf += 4;
        len -= 4;
-       ERR_FAIL_COND_V((int)count > len, ERR_INVALID_DATA);


        PoolVector<uint8_t> data;

        if (count) {
                data.resize(count);
                PoolVector<uint8_t>::Write w = data.write();
-               for (uint32_t i = 0; i < count; i++) {

                        w[i] = buf[i];
                }
```

Descoberta e Divulgação

- Descoberto por Fabio Alessandri (@Faless) ocasionalmente.

- Divulgado somente após os problemas terem sido consertados. “Divulgação responsável” é o caminho correto nestes casos, segundo o desenvolvedor.

Demonstração

- Será demonstrado somente um dos problemas mencionados: o *packet of death*.

- Foi feito um pequeno “jogo” para demonstrar este problema, que será rodado na versão 3.0.0 da Godot.

A Correção

Código de Serialização

```
                case Variant::POOL_BYTE_ARRAY: {

                        ERR_FAIL_COND_V(len < 4, ERR_INVALID_DATA);
+                       int32_t count = decode_uint32(buf);
                        buf += 4;
                        len -= 4;
+                       ERR_FAIL_COND_V(count < 0 || count > len,
ERR_INVALID_DATA);

                        PoolVector<uint8_t> data;

                        if (count) {
                                data.resize(count);
                                PoolVector<uint8_t>::Write w = data.write();
+                               for (int32_t i = 0; i < count; i++) {

                                        w[i] = buf[i];
                                }
```

Demonstração

- Retornamos, então, ao mesmo “jogo”, com o mesmo código-fonte, mas agora rodando na versão 3.0.6 da Godot.

- Esta foi a versão na qual foi divulgada a existência e resolução dos problemas mencionados.

Ressalvas

- Por mais que o *packet of death* não consiga fechar o jogo, ele gera um erro e passa a responsabilidade para a *engine*, tornando o problema de outra natureza (auto-contido).

- Note que a Godot não possui nenhuma forma de error handling.

Bibliografia

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-1000224
- https://github.com/godotengine/godot/commit/5262d1bbcc81a06db66ac45c3f75535f231268bc
- https://godotengine.org/
- https://webchat.freenode.net/?channels=#godotengine
- https://github.com/LuizGyro/mac0352-2019/tree/master/ep4-VictorAraujo-LuizGirotto

Obrigado!